1. A method comprising:

at a first point in a program in a computer programming language having dynamic types and overloaded functions, constructing , using a function name, a function data structure; the function data structure comprising information leading to a set of functions visible at the first point;

at a second point, applying the function data structure to an argument list, the applying comprising selecting a function using the function data structure and calling the selected function.

2. The method of claim 1 wherein the constructing occurs within a first scope, and wherein the applying the function data structure comprises selecting, from the set of functions led to by the function data structure, the function that would be selected if the function name were applied within the first scope to the argument list.

3. The method of claim 2 wherein the function data structure is applied in a second scope that is different from the first scope.

4. The method of claim 1 wherein the information further comprises a pointer which leads to native code.

5. The method of claim 4, wherein the information further comprises information used by the native code.

6. The method of claim 1 wherein the information further comprises a pointer which leads to native code which implements a top level function.

7. The method of claim 1 wherein the information further comprises a pointer which leads to a mapping associating a type with native code which implements a function.

8. The method of claim 1, wherein the information further comprises a pointer leading to a lexical context, and further comprises a pointer leading to native code which implements the function.

9. The method of claim 1, wherein the information further comprises a pointer leading to interpreter code which implements the function.

10. The method of claim 1, wherein the information further comprises a pointer which leads to a mapping associating a type with interpreter code for the function.

11. The method of claim 1, wherein the information further comprises a pointer leading to a lexical context, and further comprises a pointer leading to interpreter code which implements the function.

12. The method of claim 1, wherein the information further comprises information which leads to the function name.

13. The method of claim 1 in which the information further comprises information leading to an auxiliary function.

14. The method of claim 13 wherein the auxiliary function is selected from the set consisting copy, delete, print and equal.

15. The method of claim 1, wherein the information further comprises information used for storage management of the function data structure.

16. The method of claim 1 wherein the symbol "@" in front of the function name in the computer programming language means to construct the function data structure.

17. The method of claim 1 wherein the computer programming language comprises a function named "feval" which, if applied to the function data structure and the argument list, applies the function data structure to the argument list.

18. The method of claim 1, wherein the information further comprises a pointer leading to a first auxiliary function;

applying the first auxiliary function to the function data structure and obtaining a result from the applying.

19. The method of claim 18 wherein the first auxiliary function comprises at least one selected from the set consisting of copy, delete, print and equal.

20. The method of claim 18 wherein the first auxiliary function comprises a write function; and wherein applying the write function causes storing a first absolute filename of a function lead to by the information in the function data structure.

21. The method of claim 20 wherein the information further comprises a pointer leading to a read function and wherein applying the read function causes the first absolute filename to be matched with a second absolute filename on the current path which has the longest common tail with the first absolute filename.

22. The method of claim 18 wherein the information further comprises a pointer leading to a second auxiliary function, and wherein applying the second auxiliary function to the function data structure causes at least a portion of the information contained in or pointed to by the function data structure to be returned as at least one value.

23. The method of claim 1 wherein if the set of functions is other than a null set, each member of the set of functions has the same name as the function name.

24. A computer program product, stored in a computer readable medium, comprising instructions to cause a computer to:

at a first point, construct a function data structure using a function name; the function data structure comprising information leading to a set of functions visible at the first point;

at a second point, apply the function data structure to an argument list selecting a function from the function data structure and call the selected function.

25. The computer program product of claim 24 wherein the construction occurs within a first scope, and wherein the application of the function data structure comprises selecting, from the set of functions led to by the function data structure, the function that would be selected if the function name were applied within the first scope to the argument list.

26. The computer program product of claim 25 wherein the function data structure is applied in a second scope that is different from the first scope.

27. The computer program product of claim 24 wherein the information further comprises a pointer which leads to native code.

28. The computer program product of claim 27, wherein the information further comprises information used by the native code.

29. The computer program product of claim 24 wherein the information further comprises a pointer which leads to native code which implements a top level function.

30. The computer program product of claim 24 wherein the information further comprises a pointer which leads to a mapping associating a type with native code which implements a function.

31. The computer program product of claim 24, wherein the information further comprises a pointer leading to a lexical context, and further comprises a pointer leading to native code which implements the function using the lexical context.

32. The computer program product of claim 24, wherein the information further comprises a pointer leading to interpreter code which implements the function.

33. The computer program product of claim 24, wherein the information further comprises a pointer which leads to a mapping associating a type with interpreter code for the function.

34. The computer program product of claim 24, wherein the information further comprises a pointer leading to a lexical context, and further comprises a pointer leading to interpreter code which implements the function.

35. The computer program product of claim 24, wherein the information further comprises information which leads to the function name.

36. The computer program product of claim 24 in which the information further comprises information leading to an auxiliary function.

37. The computer program product of claim 36 wherein the auxiliary function is selected from the set consisting copy, delete, print and equal.

38. The computer program product of claim 24, wherein the information further comprises information used for storage management of the function data structure.

39. The computer program product of claim 24 wherein the symbol "@" in front of the function name in the instructions causes the computer to construct the function data structure.

40. The computer program product of claim 24 wherein the computer programming language comprises a function named "feval" which, if applied to the function data structure and the argument list, causes the function data structure to be applied to the argument list.

41. The computer program product of claim 24 wherein the information further comprises a pointer leading to a first auxiliary function;

and wherein the instructions cause the first auxiliary function to be applied to the function data structure and obtain a result from the applying.

42. The computer program product of claim 41 wherein the first auxiliary function comprises at least one selected from the set consisting of copy, delete, print and equal.

43. The computer program product of claim 41 wherein the first auxiliary function comprises a write function; and wherein application of the write function causes storing data on an external medium, the stored data comprising a first absolute filename of a function lead to by the information in the function data structure.

44. The computer program product of claim 43 wherein the information further comprises information leading to a read function, and wherein applying the read function to the stored data causes the first absolute filename to be matched with a second absolute filename on the current path which has the longest common tail with the first absolute filename.

45. The computer program product of claim 41 wherein the information further comprises a pointer leading to a second auxiliary function, and wherein the instructions further cause a computer to, when applying the second auxiliary function to the function data structure, cause at least a portion of the information contained in or pointed to by the function data structure to be returned as at least one value.

46. The computer program product of claim 24 wherein if the set of functions is other than a null set, each member of the set has the same name as the function name.